(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平8－69417

(43)公開日　平成８年(1996)３月12日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/12		Ａ　7623－5Ｂ		

審査請求　未請求　請求項の数２　ＯＬ　（全 10 頁）

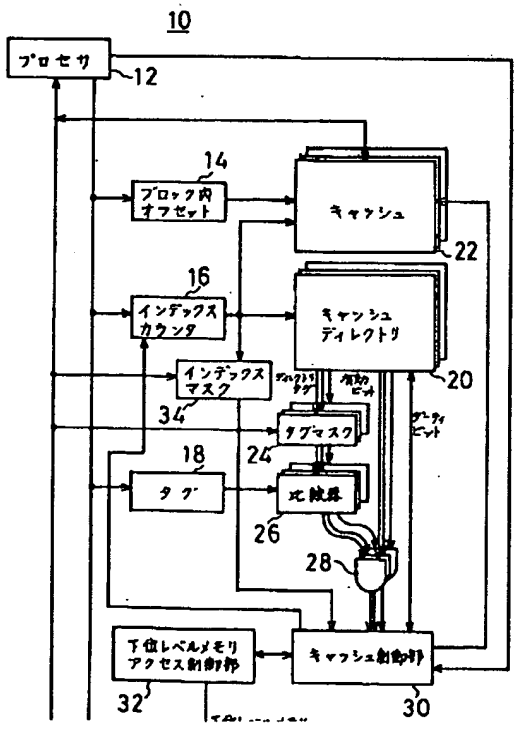
(21)出願番号　特願平6－203253

(22)出願日　平成６年(1994)８月29日

(71)出願人　000001889
三洋電機株式会社
大阪府守口市京阪本通２丁目５番５号
(72)発明者　甲村　康人
大阪府守口市京阪本通２丁目５番５号　三洋電機株式会社内
(74)代理人　弁理士　山田　義人

(54)【発明の名称】　計算機システム

(57)【要約】
【構成】　プロセサ１２の要求に従い指定されたアドレス範囲に対応するキャッシュブロックが、キャッシュディレクトリ２０からのディレクトリタグとプロセサ１２からのタグとを比較器２６で比較することによって検出する。このとき、ディレクトリタグをタグマスク２２によってマスクすることによって効率的にキャッシュブロックを検出できる。また、プロセサ１２から出力されたインデックスの値をインデックスカウンタ１６によってインクリメントし、インデックスマスク３４によってアドレス範囲に応じてインデックスカウンタ１６の上限を検出する。インデックスカウンタ１６の上限が検出されるまでの間、プロセサ１２によって指定されたアドレス範囲に対応するキャッシュブロックのダーティビットをクリアする。
【効果】　下位レベルのメモリブロックに対する不要の書き出しを防ぐことができ、計算機システムの性能が向上する。



1

【特許請求の範囲】

【請求項1】下位メモリと前記下位メモリの内容のコピーをブロック単位でもつキャッシュとを含み、前記キャッシュはプロセサからのライトアクセス要求に対してはライトバック方式によってこれを処理する計算機システムにおいて、

前記プロセサからの要求に応じて、指定されたメモリ領域に対応するキャッシュブロックのダーティビットを強制的にクリアするクリア手段を備えることを特徴とする、計算機システム。

【請求項2】前記クリア手段は、前記プロセサによって指定されたアドレス範囲に対応するキャッシュブロックのダーティビットをクリアするキャッシュ制御手段、前記アドレス範囲に対応する前記キャッシュブロックをタグマスクを用いて検出するブロック検出手段、前記プロセサから出力されるインデックスの値をインクリメントするインデックスカウンタ、および前記アドレス範囲に応じて前記インデックスカウンタの上限を検出するインデックス検出手段を備え、

前記インデックス検出手段で前記インデックスカウンタの上限が検出されるまでの間前記アドレス範囲に対応する前記キャッシュブロックのダーティビットを前記キャッシュ制御手段によってクリアする、請求項1記載の計算機システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】この発明は計算機システムに関し、特にたとえばキャッシュメモリの管理方式にライトバック方式を採用する、計算機システムに関する。

【0002】

【従来の技術】近年の高性能の計算機システムには、キャッシュと呼ばれる上位レベルの高速メモリに下位レベルのメモリの内容の一部をコピーしてもつことによってシステムの性能を向上させる機構が、広く用いられている。キャッシュの管理方式の1つは、プロセサからメモリシステムへの書き込みをキャッシュおよび下位メモリの両方に対して行い、キャッシュの内容を常に下位レベルのメモリの内容と一貫させておくライトスルー方式であり、別の1つは、プロセサからメモリシステムへの書き込みをキャッシュに対してのみ行い、データが変更されたキャッシュの内容はリプレースの対象になった時点で下位レベルのメモリに書き出されるライトバック方式である。

【0003】ライトバック方式では、キャッシュブロックの内容がプロセサによって変更されたため下位レベルのメモリの内容との一貫性を失っている状態をダーティであるといい、キャッシュブロックのそれぞれについて、ダーティか否かを示すためのダーティビットと呼ばれる状態ビットが設けられている。たとえば、従来のライ

2

式の計算機システム1は図5のように構成される。

【0004】図5を参照して、従来の計算機システム1において、プロセサ2から階層メモリに対するリードアクセス要求が出されたとする。プロセサ2から出力されたアドレスはブロック内オフセット，インデックスおよびタグに分割して保持され、インデックスで示されるキャッシュディレクトリ3のエントリがキャッシュ4の連想度の数だけ並列にアクセスされ、キャッシュディレクトリ3からディレクトリタグ，有効ビットおよびダーティビットが出力される。ディレクトリタグとアドレスのタグとは比較器5によって比較され、これらが一致し、かつ有効ビットが真であるディレクトリエントリが存在すれば、リードアクセスはキャッシュ4にヒットしたことになり、キャッシュ4がインデックスおよびプロセサ内オフセットによってアクセスされ、データがプロセサ2に返される。

【0005】もし、インデックスで示されるセット内の全てのエントリについて、有効ビットが偽であるかまたはディレクトリタグがアドレスのタグと一致しないのであれば、キャッシュミスが発生したことになる。この場合、キャッシュ制御部6はキャッシュ4中のどのキャッシュブロックを今回アクセスのあったメモリブロックの内容でリプレースするかを選択する必要がある。

【0006】リプレースの対象となったキャッシュブロックの有効ビットが偽であるか、ダーティビットが偽であるなら、そのキャッシュブロックの内容は不要あるいは下位レベルのメモリ中に同じ内容のものが存在するから、捨ててしまってかまわない。このときは、下位レベルメモリアクセス制御部7は、プロセサ2からのアクセス要求のあったアドレスを含むメモリブロックのリード要求（下位レベルメモリ制御信号）を下位レベルのメモリに対して発生する。そして、キャッシュ4のいずれかのセットが選択され、そのインデックスによって示されるエントリにこのメモリブロックの内容が転送される。また、そのメモリブロックのうちプロセサ2から要求のあったデータワードがプロセサ2にも返される。さらに、選択されたエントリのタグ部には今回アクセスされたアドレスのタグ部が設定され、有効ビットは真に、ダーティビットは偽に設定される。

【0007】一方、リプレースの対象となったキャッシュブロックの有効ビットが真であり、ダーティビットが真であるなら、キャッシュ制御部6はこのキャッシュブロックの内容をまず下位レベルのメモリに書き出して、その後にキャッシュブロックに新たな内容を読み込み上述の操作を行う必要がある。ところで、現在の典型的なプログラム、特に大規模な処理を行うプログラムにおいては、必要になった時点で生成されて不要になれば消去されるような動的な変数が多用される。このような動的変数の生成から消去までの期間を変数の生存期間と呼

的変数が最初に必要とされる時点でメモリの未使用領域から割り当てられ、その動的変数が不要になった時点で割り当てられていたメモリ領域は未使用領域に返却される。

【0008】動的変数の種類の1つは、プログラムの変数コール／リターンに伴って生成／消去されるものであり、関数の局所変数と呼ばれる。この種の動的変数はスタック領域と呼ばれるメモリ領域に割り当てられる。現在の多くのプロセサにおいては、スタック領域を管理するためのハードウェア機構を備えている。また、個々の関数呼び出しに対応して確保されるスタック領域をスタックフレームと呼ぶ。スタックフレームは、関数の局所変数，関数にわたされる引数に関する情報，および関数からのリターンアドレスなどを含む。

【0009】動的変数の別の種類は、その生存期間がプログラムの関数呼び出しと関連づけることができないものである。このような動的変数はヒープ領域と呼ばれるメモリ領域に割り当てられる。通常、この種の動的変数に対するメモリ領域の割り当ての管理は、ソフトウェアによって行われる。

【0010】

【発明が解決しようとする課題】ここで、図5に示すライトバック方式のキャッシュ4をもつ既存の計算機システム1について考える。キャッシュ4上に動的変数に対応するキャッシュブロックが存在し、なおかつこの動的変数の生存期間が終了した時点で、そのキャッシュブロックがダーティである場合を考える。この時点で、このキャッシュブロックが対応する下位レベルのメモリ領域は、ヒープ領域あるいはスタック領域であり、いずれの動的変数も割り当てられていない状態である。さらに処理が進み、このキャッシュブロックに対応するメモリ領域に新たな動的変数が割り当てられるより以前に、このキャッシュブロックがリプレースの対象になったとする。このキャッシュブロックはダーティであるため、下位レベルのメモリに内容の書き出しが行われる。

【0011】しかし、このキャッシュブロックの内容は、生存期間が終了した動的変数の内容であり、もはやプログラムによって必要とされない。すなわち、これまでの計算機システムは、生存期間の終了した動的変数がキャッシュブロック上にダーティブロックとして存在しかつそのキャッシュブロックがリプレースの対象となったときに、下位レベルのメモリにキャッシュブロックの内容を書き出すという不要な操作を行っていた。

【0012】それゆえに、この発明の主たる目的は、不要な操作を防ぎ性能を向上することができる、計算機システムを提供することである。

【0013】

【課題を解決するための手段】この発明は、下位メモリと下位メモリの内容のコピーをブロック単位でもつキャッシュを含み、キャッシュはプロセサからのライトバック方式によってこれを処理する計算機システムにおいて、プロセサからの要求に応じて、指定されたメモリ領域に対応するキャッシュブロックのダーティビットを強制的にクリアするクリア手段を備えることを特徴とする、計算機システムである。

【0014】

【作用】プロセサからの要求に従い、指定されたアドレス範囲に対応するキャッシュブロックをブロック検出手段でタグマスクを用いて効率よく検出する。また、インデックスカウンタによって、プロセサから出力されたインデックスの値をインクリメントし、インデックス検出手段で、指定されたアドレス範囲に応じてインデックスカウンタの上限を検出する。そして、インデックスカウンタの上限が検出されるまでの間、プロセサによって指定されたアドレス範囲に対応するキャッシュブロックのダーティビットを、キャッシュ制御手段によって強制的にクリアする。

【0015】ここで、強制的なダーティビットのクリアは、キャッシュブロックの内容が下位レベルのメモリブロックとの同一性を失っているにも拘わらず、その情報が計算機システムから失われてしまう、すなわち、計算機システムが一貫性を失ってしまうことを意味する。このことが正しいプログラムの実行に影響を与えないことを以下に示す。

【0016】ここでは、動的変数の生存期間が終了した場合を例に説明する。一般に、動的変数が割り当てられているメモリ領域は、プログラム実行時に決定され、どのメモリ領域が割り当てられているか、さらに、割り当てられた領域に偶然格納されている値がどのようなものがあるか、プログラマーが事前に知ることはできない。すなわち、動的変数の初期値はプログラマーにとっては非決定的であると考えられている。したがって、正しいプログラムは動的変数の初期値に依存しないようにコーティングされている。

【0017】ここで、アドレスaに割り当てられた動的変数の生存期間が終了し、アドレスaに対応するキャッシュブロックcのダーティビットが強制的にクリアされたとする。この時点以降、キャッシュブロックcに対する最初の操作は、読み出し、書き込み、リプレースのいずれかである。キャッシュブロックcに対して読み出しが実行される場合は、アドレスaに新たな動的変数が割り当てられ、それが参照される場合のみである。しかし、上述のように動的変数の初期値に依存するプログラムは正しいプログラムとはいえないため、この場合を考慮する必要はない。

【0018】キャッシュブロックcに対して書き込みが実行される場合は、アドレスaに新たな動的変数が割り当てられ、それに対する代入が行われた場合である。こ

なる。これにより、計算機システムの一貫性は回復され、先に実行されたダーティビットのクリア操作が、これ以降の実行に影響を及ぼすことはない。

【0019】キャッシュブロックcに対してリプレースが実行された場合、プログラムから見たアドレスaの内容は、下位レベルのメモリブロックの内容となる。これはプロセサが最後（リプレース直前）にアドレスaに対して書き込んだ内容、すなわちキャッシュブロックcの内容とは異なる。しかし、アドレスaは動的変数のための領域すなわちスタック領域あるいはヒープ領域であり、アドレスaには動的変数が割り当てられていない状態である。将来、プロセサがアドレスaのメモリ領域を利用する場合は、新たな動的変数がアドレスaに割り当てられることを意味するが、上述のように正しいプログラムは動的変数の初期値に依存しないように設計されているため、アドレスaのメモリブロックに格納されている値はどのような値でもプログラムの実行に影響を及ぼすことはない。したがって、先に実行されたダーティビットクリア操作が、これ以降の実行に影響を及ぼすことはない。

【0020】

【発明の効果】この発明によれば、ダーティなキャッシュブロックがリプレースの対象となった場合に、キャッシュブロックのダーティビットを強制的にクリアすることによって、正しいプログラムの実行に影響を与えることなく下位レベルのメモリブロックに対する不要な書き出しを防ぐことができ、ひいては計算機システムの性能を向上させることができる。

【0021】この発明の上述の目的，その他の目的，特徴および利点は、図面を参照して行う以下の実施例の詳細な説明から一層明らかとなろう。

【0022】

【実施例】図1を参照して、この実施例の計算機システム10は、たとえば、ライトバック方式でありかつセットアソシアティブキャッシュ方式に構成されたものである。計算機システム10は、プロセサ12を含む。プロセサ12から階層メモリシステムに対するアクセス要求のあったアドレスは、ブロック内オフセット，インデックスおよびタグに分割され、それぞれブロック内オフセット用レジスタ14，インデックスカウンタ16およびタグ用のレジスタ18に保持される。そして、インデックスで示されるキャッシュディレクトリ20のエントリがキャッシュ22の連想度の数だけ並列にアクセスされ、キャッシュディレクトリ20からディレクトリタグ，有効ビットおよびダーティビットが出力される。ディレクトリタグはタグマスク24によってマスクされた後、レジスタ18からのアドレスのタグと比較器26によって比較され、これらが一致し、かつ有効ビットが真であるディレクトリエントリすなわちキャッシュブロックが存在すれば、ヒットとなる。

ットしたことになり、ANDゲート28を介してその旨の信号がキャッシュ制御部30に与えられる。すると、キャッシュ制御部30によって、キャッシュ22がインデックスおよびプロセサ内オフセットに基づいてアクセスされ、データがプロセサ12に返される。

【0023】もし、インデックスで示されるセット内の全てのエントリについて、有効ビットが偽であるかまたはディレクトリタグがアドレスのタグと一致しないのであれば、キャッシュミスが発生したことになる。この場合、図5に示す従来技術と同様、キャッシュ制御部30は、キャッシュ22中のどのキャッシュブロックを今回アクセスのあったメモリブロックの内容でリプレースするかを選択する必要がある。

【0024】すなわち、リプレースの対象となったキャッシュブロックの有効ビットが偽であるか、ダーティビットが偽であるなら、そのキャッシュブロックの内容は不要あるいは下位レベルのメモリ中に同じ内容のものが存在するから、捨ててしまってかまわない。このときは、下位レベルメモリアクセス制御部32は、プロセサ12からのアクセス要求のあったアドレスを含むメモリブロックのリード要求（下位レベルメモリ制御信号）を下位レベルのメモリ（図示せず）に対して発生する。そして、キャッシュ22のいずれかのセットが選択されて、そのインデックスによって示されるエントリにこのメモリブロックの内容が転送される。また、そのメモリブロックのうちプロセサ12から要求のあったデータワードがプロセサ12にも返される。さらに、選択されたエントリのタグ部には今回アクセスされたアドレスのタグ部が設定され、有効ビットは真に、ダーティビットは偽に設定される。

【0025】また、リプレースの対象となったキャッシュブロックの有効ビットが真であり、ダーティビットが真であるなら、キャッシュ制御部30はこのキャッシュブロックの内容をまず下位レベルのメモリに書き出して、その後にキャッシュブロックに新たな内容を読み込み上述の操作を行う。このように、計算機システム10は、図5に示す従来の計算機システム1の有する機能を備えているが、計算機システム10ではさらに以下の点に注目すべきである。

【0026】すなわち、計算機システム10では、プロセサ12からの要求に従い、指定されたアドレス範囲に対応するキャッシュブロックのダーティビットをクリアする機能をもつように改良されたキャッシュ制御部30，指定されたアドレス範囲に対応するキャッシュブロックを効率よく見出すために、ディレクトリタグをマスクするタグマスク24，インデックスの値をインクリメントするインデックスカウンタ16，およびインデックスカウンタ16の上限を与えるインデックスマスク34を備えることである。

【0027】タグマスク24は、たとえば図2に示すと

うに構成される。図２に示すタグマスク２４は、ビット毎のＮＯＴ回路２４ａを含み、データバスに現れるmask-tagをＮＯＴ回路２４ａによってビット毎に否定演算し、得られた～mask-tagをタグマスクレジスタ２４ｂに与え、後述する図４のアルゴリズムの実行中に～mask-tagはタグマスクレジスタ２２ｂで保持される。また、キャッシュディレクトリ２０によって生成されるディレクトリタグの値とタグマスクレジスタ２４ｂの内容とのビット毎の論理積がＡＮＤ回路２４ｃによって生成され、得られた結果が比較器２６へ転送される。このようなタグマスク２４を用いかつ後述の制約条件を満たすことによって、検査すべきキャッシュブロックの数を最小限にでき、その結果、キャッシュブロックの検出を効率的に行える。

【００２８】また、インデックスマスク３４は、たとえば図３に示すように構成される。図３に示すインデックスマスク３４は、インデックスマスクレジスタ３４ａを含む。インデックスマスクレジスタ３４ａは、図４に示すアルゴリズムを実行する間、mask-idxを保持しておく。インデックスマスクレジスタ３４ａからの値とインデックスカウンタ１６によって生成されるインデックス値とのビット毎の論理積がＡＮＤ回路３４ｂによって生成される。さらに、ＡＮＤ回路３４ｂによって生成される値とインデックスマスクレジスタ３４ａに保持される値が比較器３４ｃによって比較され、それらの値が等しいときには、比較器３４ｃからキャッシュ制御部３０に図４に示すアルゴリズムの終了が通知される。すなわち、インデックスマスク３４によって、プロセサ１２から指示されたアドレス範囲に対応してインデックスカウンタ１６の上限を検出している。

【００２９】図１に戻って、さらに、プロセサ１２は、ユーザプログラムにて実行可能な、特定のアドレス範囲に対応するキャッシュブロックのダーティビットをクリアするための命令

clean-cache　　addr mask

をもつ。この命令は、たとえば、動的変数の生存期間が終了した時点でプロセサ１２から出力され、キャッシュ制御部３０に入力される。またこの命令は、２のべき乗の値をとるアドレスaddrと２のべき乗－１の値をとるmaskとをオペランドとし、addrからaddr＋maskまでのアドレスに対応するキャッシュブロックを全て見出し、そのキャッシュブロックのダーティビットをクリアする操作をキャッシュ制御部３０に指示する。ただし、mask＋１はキャッシュ２２のブロックサイズ以上でなければならず、addrとmaskとのビット毎の論理積をとったものは０に等しいという制約条件を満たすものとする。

【００３０】キャッシュ制御部３０は、上述のプロセサ１２からの命令の要求に応じて、図４に示す手順に従ってキャッシュブロックのサーチおよびダーティビットクリアの操作を行う。ここで、clean-cache 命令のオペランドであるaddrはアドレスバスに、maskはデータバスにそれぞれプロセサ１２から出力されるものとする。図４に示す動作において、まず、clean-cache 命令に与えられたaddrおよびmaskのそれぞれのタグ部をaddr-tagおよびmask-tagとし、addrおよびmaskのそれぞれのインデックス部をaddr-idxおよびmask-idxとする。

【００３１】そして、ステップＳ１において、addr-tagをタグ用のレジスタ１８に、～mask-tagをタグマスクレジスタ２４ｂに、addr-idxをインデックスカウンタ１６に、mask-idxをインデックスマスク３４に、それぞれ保持する。その後、キャッシュ２２の各セットについて並列にステップＳ３およびＳ５をそれぞれ実行する。ステップＳ３において、ディレクトリタグ&タグマスクレジスタ２４ｂの出力（～mask-tag）＝タグ用のレジスタ１８の出力（addr-tag）であるか否かが判断される。これは、タグマスク２４および比較器２６で行われる。これによって、キャッシュ２２内の或るキャッシュブロックがプロセサ１２から要求されているアドレス範囲のキャッシュブロックに相当するか否かが判断される。ステップＳ３が“ＹＥＳ”であれば、ステップＳ５に進む。なお、このとき、有効ビットは真であることを要する。ステップＳ５において、インデックスカウンタ１６で示されるディレクトリエントリすなわちキャッシュブロックのダーティビットがクリアされ、ステップＳ７に進む。ステップＳ３が“ＮＯ”のときは直接ステップＳ７に進む。ステップＳ７において、インデックスカウンタ１６がインクリメントされ、ステップＳ９に進む。ステップＳ９では、インデックスカウンタ１６の出力（addr-idx）&インデックスマスクレジスタ３４ａの出力（mask-idx）＝インデックスマスクレジスタ３４ａの出力（mask-idx）であるか否かが判断される。これによってインデックスカウンタ１６の上限を判断することができる。ステップＳ９が“ＮＯ”であればインデックスカウンタ１６の出力はまだインデックスカウンタ１６の上限ではないと判断され、上述の処理が繰り返される。ステップＳ９が“ＹＥＳ”であれば、インデックスカウンタ１６の出力がインデックスカウンタ１６の上限になったと判断され、終了する。

【００３２】さらに具体的に説明する。ここで、キャッシュ２２のブロックサイズをblocksize とし、キャッシュ２２のセット数をsetnumとする。また、addr，mask＋１，blocksize ，setnumは、それぞれ２のべき乗の値であり、定義より数１が成立する。

【００３３】

【数１】addr-tag＝addr／setnum／blocksize

addr-idx＝（addr／blocksize）％ setnum

mask-tag＝mask／setnum／blocksize

mask-idx＝（mask／blocksize）％ setnum

数１において、％ はモジュロ演算を表している。ここ

9

つblock-idx なるインデックスに対応するダーティなキャッシュブロックが存在したとする。このキャッシュブロックが対応付けられているメモリ領域〔block-min … block-max〕は、数2によって表される。

【0034】

【数2】block-min ＝（block-tag ＊setnum＋block-idx）＊ blocksize

block-max ＝（block-tag ＊setnum＋block-idx ＋1）＊blocksize －1

ここで、図4に示すアルゴリズムによって、〔block-min … block-max〕が〔addr…addr＋mask〕に含まれるとき、また、そのときに限りこのキャッシュブロックのダーティビットがクリアされることを示す。

【0035】図4のアルゴリズムにおいて、ステップS3を実行するときのインデックスカウンタ16の値idxは、〔addr-idx…addr-idx＋mask-idx〕の全ての値をとる。なぜなら、ステップS1によってidx の初期値はaddr-idxであり、制約条件よりaddr-idx&mask-idx＝0であるから、ステップS7にてインクリメントされた値idx において、ステップS9でidx &mask-idx＝mask-idxが最初に成立するのはidx ＝addr-idx＋mask-idxの場合となるためである。

【0036】すなわち、ステップS9が“YES”になり終了するのは、idx ＝addr-idx＋mask-idxの場合だからである。以下、mask＋1とsetnum＊blocksize の大小関係によって、mask＋1＜setnum＊blocksize の場合と、mask＋1≧setnum＊blocksize の場合の2通りに場合分けして考える。

【0037】(1) mask＋1＜setnum＊blocksize の場合

mask＋1＜setnum＊blocksize であるから、mask-tag＝0が成立する。したがって、〔block-min … block-max〕が〔addr…addr＋mask〕に含まれるということはblock-idx が〔addr-idx…addr-idx＋mask-idx〕に含まれ、かつblock-tag＝addr-tagであるということに他ならない。したがって、図4に示すアルゴリズムにおいて、ステップS3で検査されるキャッシュブロックは、そのblock-idxが〔addr-idx…addr-idx＋mask-idx〕に含まれるもの全てであり、それ以外にない。また、mask-tag＝0であるから、ステップS3の検査はblock-tag＝addr-tagであるかどうかの検査と等価である。

【0038】(2) mask＋1≧setnum＊blocksize の場合

この場合、addr-idx＝0かつmask-idx＝setnum－1が成立する。すなわち、図4のアルゴリズムにおいて、ステップS3で検査されるキャッシュブロックはキャッシュ22中の全てのキャッシュブロックとなる。ここで、〔block-min … block-max〕が〔addr…addr＋mask〕に含まれるということはblock-tag が〔addr-tag…addr-tag＋mask-tag〕に含まれるということに他ならない。なぜなら、block-idx は必ず〔addr-idx…addr-idx＋mask-idx〕すなわち〔0…setnum－1〕に含まれるためである。

10

【0039】ここで制約条件より、addr-tag&mask-tag＝0であるから、ステップS3で検査されるblock-tag&～mask-tag＝addr-tagであるかどうかは、block-tagが〔addr-tag…addr-tag＋mask-tag〕に含まれるかどうかということと等価である。以上のように、図4に示すアルゴリズムでは、〔block-min … block-max〕が〔addr…addr＋mask〕に含まれるキャッシュブロックについては必ずステップS5が実行され、それ以外のキャッシュブロックについてはステップS5は実行されないことがわかる。

【0040】この実施例によれば、たとえば、生存期間が終了した動的変数が置かれたダーティなキャッシュブロックがリプレースの対象となった場合に、そのキャッシュブロックのダーティビットをクリアすることによって不要な操作を防ぎ、計算機システム10の性能を向上させることができる。なお、上述の実施例では、clean-cache 命令に与えることのできるアドレス範囲に制限を設けていた。しかし、別の構成例として、より多くのハードウェア量を必要とするが、それぞれアドレス範囲の上限および下限と比較するための2個の大小比較器をタグ部の比較に用いることによって、任意のアドレス範囲をclean-cache 命令に与えることができる計算機システムを構成することもできる。

【0041】また、上述の実施例では、セットアソシアティブ方式のキャッシュをとりあげて説明したが、ダイレクトマップ方式あるいはフルアソシアティブ方式はセットアソシアティブ方式の極端な場合であると捉えることができるため、ダイレクトマップ方式あるいはフルアソシアティブ方式にこの発明を適用することもできる。

【0042】なお、clean-cache 命令は、上述のようにたとえば、ヒープ領域およびスタック領域に割り当てられた動的変数のメモリ割り当てを解放するときに、そのメモリ領域に対して用いる。特に、スタック領域に関しては、関数のリターン時に解放されるスタックフレーム領域全体に対するclean-cache を実行することができる。しかしながら、この発明は、プログラムによって利用される動的変数の生存期間が終了した場合、すなわち動的変数のメモリ割り当てを解放する場合以外でも用いられることはいうまでもない。

【図面の簡単な説明】

【図1】この発明の一実施例を示すブロック図である。

【図2】この実施例のタグマスクの構成の一例を示す図解図である。

【図3】この実施例のインデックスマスクの構成の一例を示す図解図である。

【図4】この実施例のキャッシュ制御部がclean-cache 命令を実行する場合のアルゴリズムを示すフロー図である。

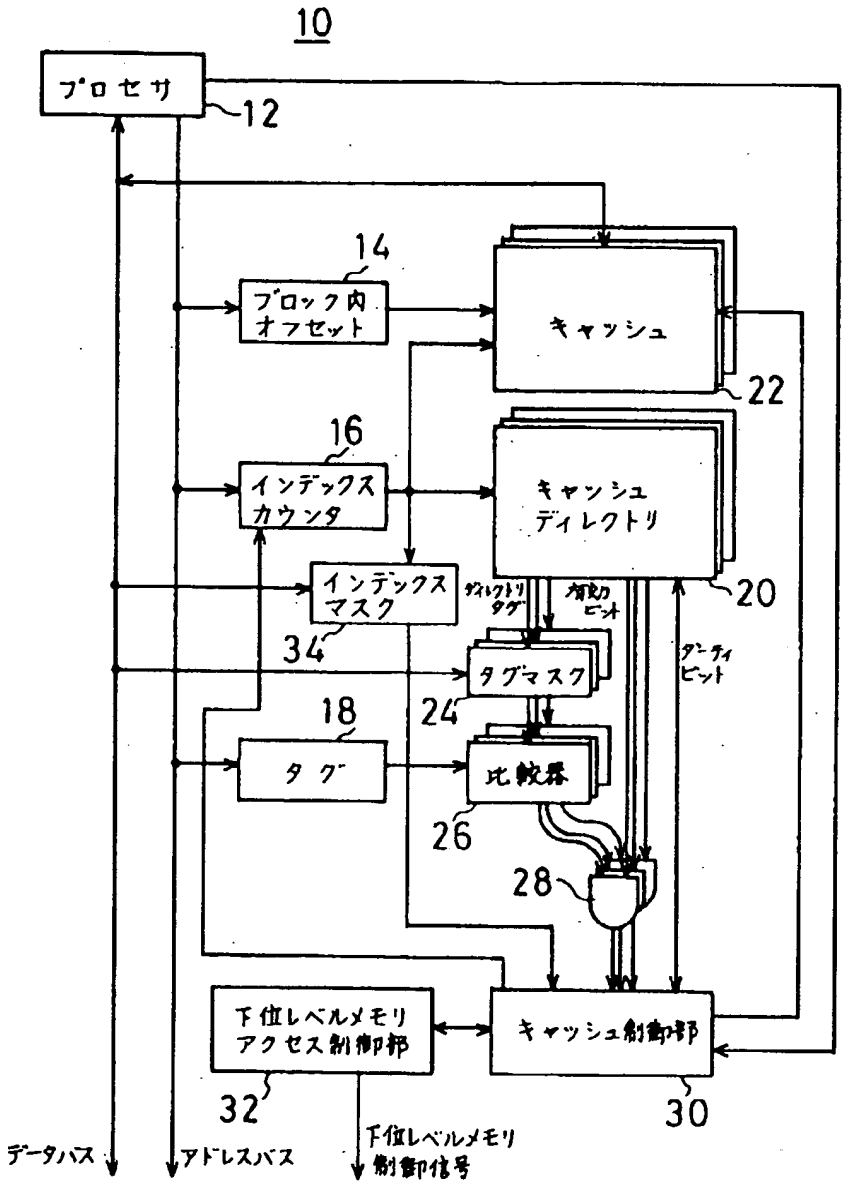
【図5】従来技術を示すブロック図である。

11　　12

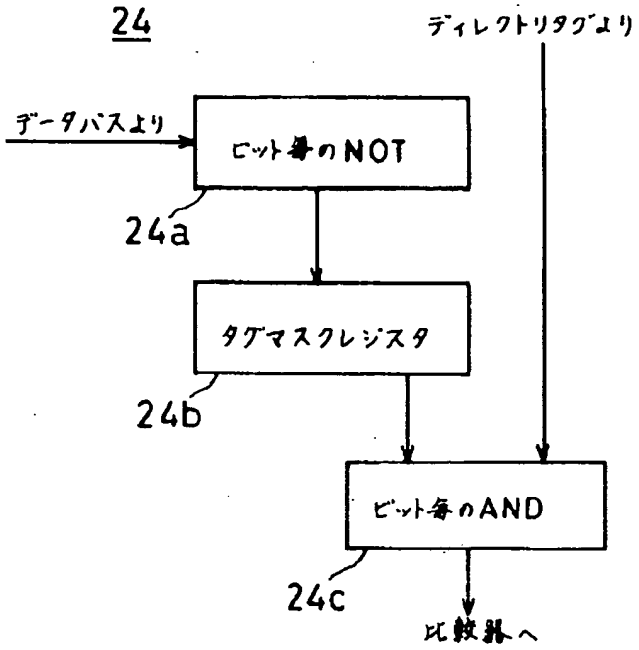
【符号の説明】

10　…計算機システム
12　…プロセサ
16　…インデックスカウンタ
20　…キャッシュディレクトリ
22　…キャッシュ
24　…タグマスク
26　…比較器
30　…キャッシュ制御部
34　…インデックスマスク

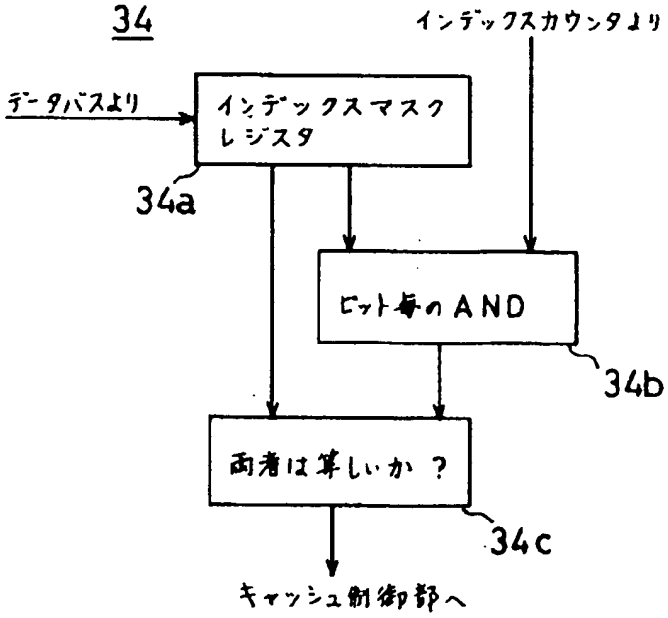
【図1】



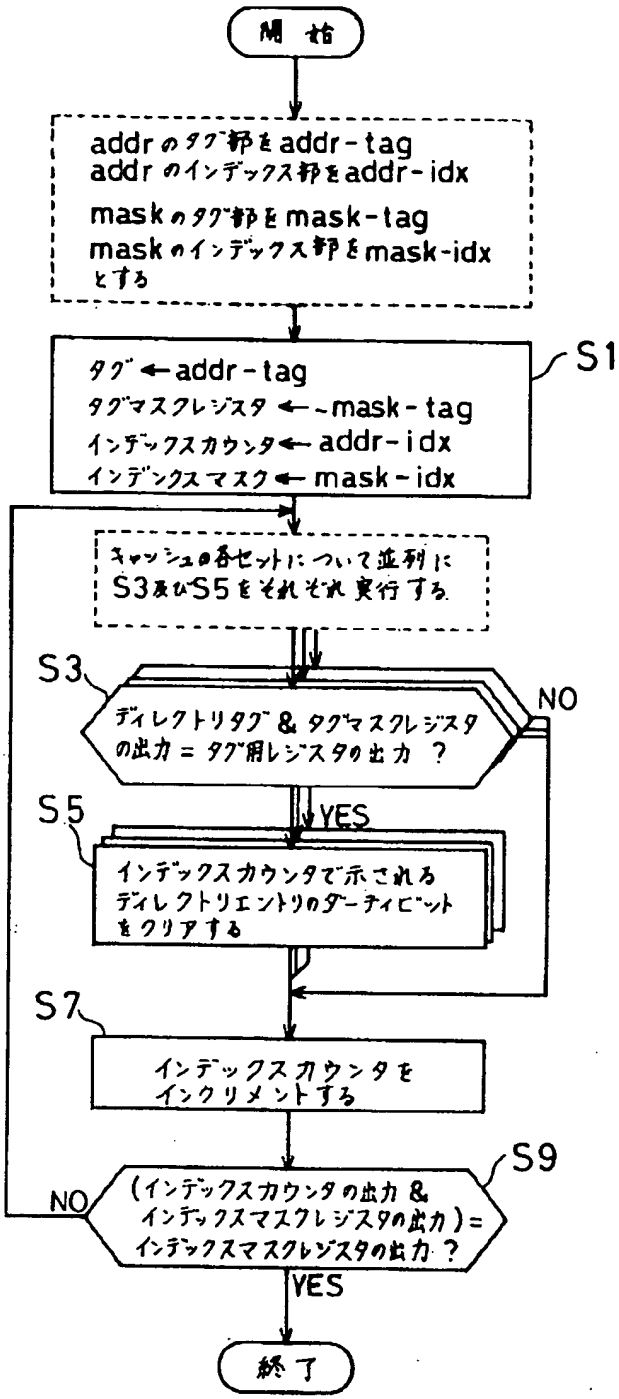
【図2】


24
ディレクトリタグより
データバスより
ビット毎のNOT
24a
タグマスクレジスタ
24b
ビット毎のAND
24c
比較器へ

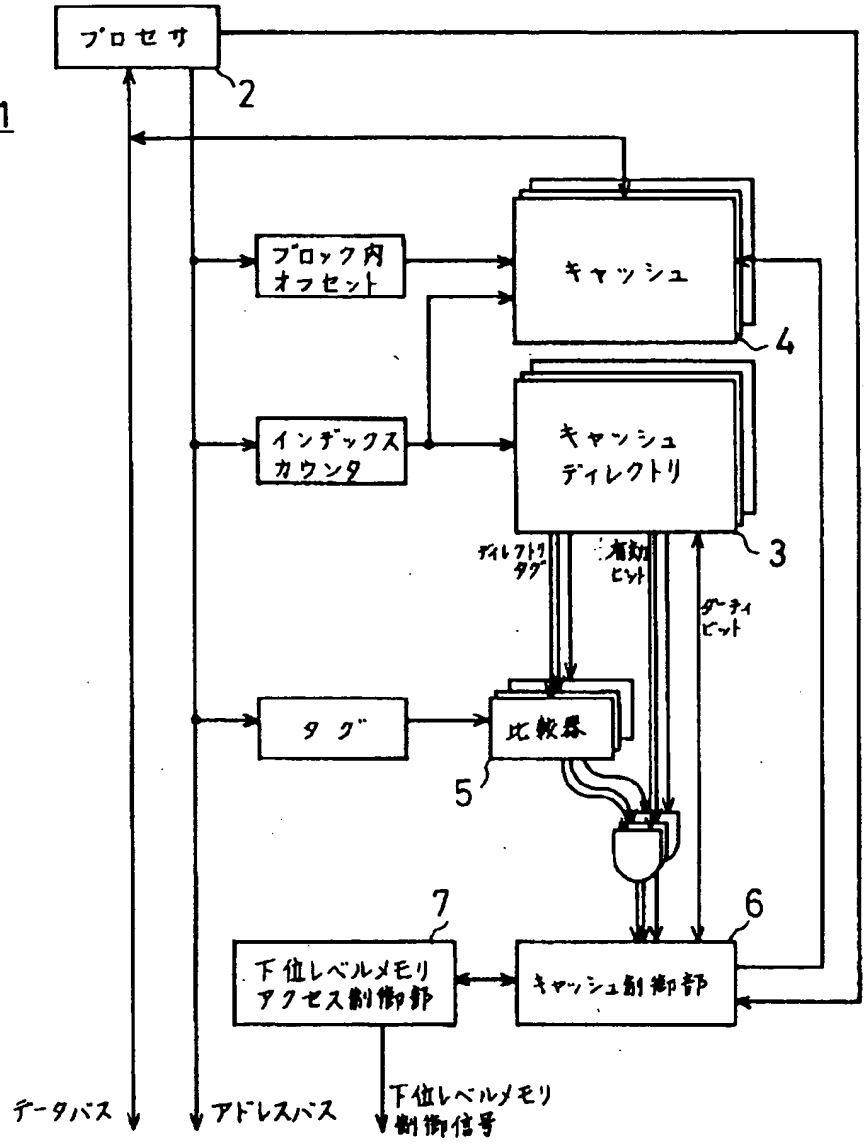
【図3】


34
インデックスカウンタより
データバスより
インデックスマスクレジスタ
34a
ビット毎のAND
34b
両者は等しいか？
34c
キャッシュ制御部へ

【図4】


開始
addrのタグ部をaddr-tag
addrのインデックス部をaddr-idx
maskのタグ部をmask-tag
maskのインデックス部をmask-idx
とする
タグ←addr-tag
タグマスクレジスタ←mask-tag
インデックスカウンタ←addr-idx
インデックスマスク←mask-idx
S1
キャッシュの各セットについて並列に
S3及びS5をそれぞれ実行する
S3
ディレクトリタグ & タグマスクレジスタ
の出力 = タグ用レジスタの出力 ？
NO
YES
S5
インデックスカウンタで示される
ディレクトリエントリのダーティビット
をクリアする
S7
インデックスカウンタを
インクリメントする
S9
(インデックスカウンタの出力 &
インデックスマスクレジスタの出力)=
インデックスマスクレジスタの出力？
NO
YES
終了

【図5】


プロセサ
2
1
ブロック内
オフセット
キャッシュ
4
インデックス
カウンタ
キャッシュ
ディレクトリ
3
ディレクトリ
タグ
有効
ビット
ダーティ
ビット
タグ
比較器
5
7
6
下位レベルメモリ
アクセス制御部
キャッシュ制御部
データバス
アドレスバス
下位レベルメモリ
制御信号